Sequence and Communication Diagrams

- Class Diagrams give a static view of the system. Time is not involved.
- We need also a way to describe the dynamics of the system.
- Sequence diagrams describe typical sequences of method calls.
- Example...
A sequence diagram

Objects

: Button
: DigitListener
: MainFrame
: Model

actionPerformed(ActionEvent)
digit(int)
refresh( )
getResult( )
getOperand( )

The Button is executing processActionEvent. In response to the user clicking on the Button.

Method call

Method Return

© 2017 T. S. Norvell, A. Vardy
Named object and Calls to self

Objects can be given names

m : MainFrame

b : Button

l : DigitListener

Repeated for each button

Button("0":String)

DigitListener(m:MainFrame, model:Model, 0:int)

addActionListener(l:ActionListener)

Calls to self

add(b:Component)

refresh( )
Show creation and deletion

- For a language with garbage collection, the big X just means the object is available for removal.
Loops and conditionals

- Sequence diagrams are not ideal for showing loops and conditionals
  - It is better to use pseudocode or other diagram (activity or state diagram)
  - Or simply describe your control flow in additional comments or documentations

- If you really want to, the following slide (from “UML Distilled”) gives an example of the notation...
The following slide gives a sequence diagram for this procedure:

```
procedure dispatch
    foreach (lineitem)
        if (product.value > $10K)
            careful.dispatch
        else
            regular.dispatch
        end if
    end for
    if (needsConfirmation) messenger.confirm
end procedure
```
procedure dispatch
  foreach (lineitem)
    if (product.value > $10K)
      careful.dispatch
    else
      regular.dispatch
    end if
  end foreach
  if (needsConfirmation) messenger.confirm
end procedure
Sequence Diagrams …

- Hint at but do not describe algorithms
  - For algorithms use pseudo code and/or activity charts and/or state diagrams
- Are good at showing typical or important interactions between a few objects
- Cut across levels of abstraction
Communication Diagrams

- Show the same information as sequence diagrams and also links between the objects
- Show the ordered tree structure of calls by hierarchical numbering